XENIX® System V

Operating System

Introduction to XENIX

This document was typeset with an IMAGEN® 8/300 Laser Printer.

SCO Document Number: XG-5-1-86-3.0

Contents

Chapter 1

Introduction

1.1 Overview

This guide introduces key concepts of the XENIX system by presenting them in a tutorial format.

It begins with a "demonstration" that explains an actual computer session, including command usage and correcting typing errors. Basic concepts such as files, commands, and pattern matching are also introduced.

Finally, these and other concepts are applied to many "real world" examples, such as file manipulation, terminal configuration, process control, and status information.

Note

This guide should be read before the other XENIX documentation; however, for more detailed discussions of all topics covered here, consult the other user's and reference guides in the set.

1.2 The XENIX System

The XENIX system consists of a general-purpose multi-user operating system and over one hundred utilities and application programs. In addition to the XENIX Operating System described in this guide, two other XENIX system packages are available: the XENIX Development System and the XENIX Text Processing System.

1.3 The XENIX Working Environment

The XENIX system is built around the XENIX operating system. The purpose of an operating system is to efficiently organize and control the resources of a computer so that they can be used by real people. These resources include memory, disks, lineprinters, terminals, and any other peripheral devices connected to the system. The heart of the XENIX system is a "multi-user" and "multi-tasking" operating system. A multi-user system permits several users to use a computer simultaneously, thus providing lower cost in computing power per user. A multi-tasking system permits several programs to run at the same time and increases productivity because multiple programs can run simultaneously rather than in sequence.

Because UNIX™ (and thus XENIX) is an accepted standard for "high-end" operating systems, a great deal of software is available for this

environment. In addition, XENIX provides file access to the MS-DOS™ operating system, the most widely used 16-bit operating system in the world. For systems that support DOS, XENIX provides commands that let you access DOS format files and disks. The XENIX system also includes several widely praised enhancements developed at the University of California at Berkeley, and a visual interface similar to other Microsoft productivity tool interfaces.

Other characteristics of the XENIX system include:

- A powerful command language for programming XENIX commands. Unlike other interactive command languages, the XENIX "shell" is a full programming language.
- Simple and consistent naming conventions. Names can be used absolutely, or relative to any directory in the file system.
- Device-independent input and output: each physical device, from interactive terminals to main memory, is treated like a file, allowing uniform file and device input and output.
- A set of related text editors, including a full screen editor.
- Flexible text processing facilities. In XENIX, commands exist to find and extract patterns of text from files, to compare and find differences between files, and to search through and compare directories. Text formatting, typesetting, and spelling error-detection facilities, as well as a facility for formatting and typesetting complex tables and equations are also available.
- A sophisticated "desk-calculator" program.
- Mountable and dismountable file systems that permit addition of floppy disks to the file system.
- A complete set of flexible directory and file protections that allows all combinations of read, write, and execute access for the owner of each file or directory, as well as for groups of users.
- Facilities for creating, accessing, moving, and processing files and directories in a simple and uniform way.

1.4 About This Guide

This guide is organized as follows:

Chapter 1, "Introduction," gives an introduction and overview of the XENIX system.

Chapter 2, "Demonstration," gives you hands-on experience in using the XENIX system.

Chapter 3, "Basic Concepts," explains the fundamental concepts that you need to understand before you begin to use the system. Included here are sections on the file system, naming conventions, commands, and input and output.

Chapter 4, "Tasks" explains how to perform everyday tasks using appropriate XENIX commands.

Chapter 2

Demonstration

2.1 Introduction

This chapter contains a demonstration run designed to help you get used to the XENIX system, so that you can quickly start to make effective use of it. It shows you how to log in, how to enter at your keyboard, what to do about mistakes in entering, how to enter commands and how to log out.

2.2 Before You Log In

Before you can log in to the system, your name must be added to the XENIX user list. At that time you will be given a login name and a password. You may have to add your name yourself, or someone else may be assigned this task; it all depends on the environment in which your system is used. In any case, see the XENIX *Operations Guide* and **mkuser**(C) for detailed information on adding users.

When you are given an account on the XENIX system you will also receive a user name, a password, and a login directory. Once you have these, all you need is a terminal from which you can log in to the system. XENIX supports most terminals and you should have no problem getting your terminal to work with XENIX. Once again, see the XENIX *Operations Guide* for more information on how to configure your terminal.

2.3 Logging In

Normally the system is sitting idle with a "login:" prompt on the terminal screen. If the system displays nonsense characters when you enter text, then your terminal is probably receiving information at the wrong speed and you should check your terminal switches. If the switches are set correctly, push the **BREAK** or **INTERRUPT** key a few times.

When you get a "login:" message, enter your login name, then press **RETURN**; the system will not do anything until you do. If a password is required, you will be asked for it. The password that you enter does not appear on the screen. This prevents others from viewing it. Do not forget to press **RETURN** after you enter your password. Next you see the line

```
TERM = (unknown)
```

Enter your terminal type (for example, ansi) and press **RETURN**.

A successful log in produces a "prompt character", a single character that indicates the system is ready to accept commands. The prompt is usually a dollar sign ($) or a percent sign (%).

You may also get a login message such as:

You have mail

telling you that another system user has sent you mail.

2.4 Typing Commands

Once the prompt character appears, the system is ready to respond to commands entered at the terminal. Try entering:

date

followed by **RETURN**. The system responds by displaying something like:

Mon Jun 16 14:17:10 EST 1985

Do not forget to press the **RETURN** key after the command, or nothing will happen. The **RETURN** key will not be mentioned again, but do not forget -- it has to be entered at the end of each command line. On some terminals **RETURN** may be labeled "ENTER" or "CR", but in all cases, the key performs the same function.

Another command you might try is **who**, which lists the names of everyone who is logged in to XENIX. A typical display from the **who** command might look something like this:

```
you     console   Jan 16   14:00
joe     tty01     Jan 16   09:11
ann     tty02     Jan 16   09:33
```

The time, given in the fourth column, indicates when the user logged in; tty*nn* is the system name for each user's terminal, where *nn* is a unique two-digit number. The console is the special name of the master terminal that is the default for most operations.

If you make a mistake entering the command name, you will see a message on your screen. For example, if you enter:

whom

the system responds with the message:

whom: not found

Note that case is significant in XENIX. The commands

who

and

WHO

are not the same; this differs from some operating systems, where case does not matter.

Now try displaying a message on your screen using the **echo** command. Type:

echo hello world

The **echo** command does what its name implies and echoes the rest of the command line to your terminal:

hello world

Now try this:

echo hello world > greeting.file

This time the **echo** command sends its output to a new file named *greeting.file*, instead of to your terminal. Note the use of the greater-than sign (>) to "redirect" the output of the command. Now enter:

ls

to list just the name of the file. To look at the contents of display it by entering:

cat greeting.file

Here "cat" stands for concatenate. One purpose of the **cat** command is to combine the contents of several files (that is, "concatenate") and put them in some new file. However, since your terminal display is treated like any other file in XENIX, **cat** is most commonly used to display the contents of files on the screen. Therefore the above command sends the following output to your terminal screen:

hello world

To remove greeting.file, enter:

rm greeting.file

Note that XENIX command names are often shortened to mnemonic names. For example, **cp** is short for "copy", **ls** is short for "list", **rm** is short for "remove", **cat** is short for "concatenate", **mkdir** is short for "make directory", and **chmod** is short for "change mode".

2.5 Mistakes in Typing

If you make a mistake in entering while entering a command, there are two ways to edit the line, provided you have not yet pressed **RETURN**. Pressing the **BKSP** key causes the last character entered to be erased. Backspacing with the **BKSP** key can erase characters back to the beginning of the line, but not beyond. Thus, if you type badly, you can correct as you go. For example, entering:

ddBKSPateRETURN

is the same as

dateRETURN

The XENIX kill character, **Ctrl-u**, erases all of the characters entered so far on the current input line. So, if the line is irretricvably fouled up, enter **Ctrl-u** and start the line over.

If you must enter a **BKSP** or **Ctrl-u** as part of the text, precede it with a backslash (\), so that the character loses its special ""erase"" meaning. To enter a **BKSP** or **Ctrl-u** in text, enter "\BKSP" or "**Ctrl-u**". The system always prints a new line on your terminal after your **Ctrl-u**, even if preceded by a backslash. Nevertheless, the **Ctrl-u** will have been recorded.

To erase a backslash, backspace twice with the BKSP key, as in "\BKSPBKSP". The backslash is used extensively in XENIX to indicate that the following character is in some way special. Note that the functions performed by **BKSP** and **Ctrl-u** are available on all XENIX systems; however, the keys used to perform these functions may vary and can be set by the user with **stty**(C).

2.6 Read-Ahead and Type-Ahead

XENIX has full read-ahead, which means that you can type as fast as you want, whenever you want, and XENIX will remember what you have entered. If you enter any text while a command is displaying text on the screen, your input characters appear intermixed with the output characters on the screen, but they are stored away and interpreted in the correct order. Therefore, you can enter several commands (i.e., "type ahead") one after another without waiting for the first to finish. Note that this does

not work when you log in; type-ahead does not work until *after* you have entered your password and the dollar sign ($) prompt appears.

2.7 Strange Terminal Behavior

Occasionally, your terminal may act strangely. You can often fix such behavior by either turning your terminal off, then quickly turning it back on, or logging out and logging back in; this will reset your terminal characteristics. It is often helpful to enter a **Ctrl-q**. This restores terminals that are (inadvertantly or otherwise) in a non-echoeing mode. **Ctrl-s** stops display to the screen, **Ctrl-q** restarts display. If logging out and back in, turning the terminal off and on, and entering **Ctrl-q** does not work, read the description of the command stty(C) in the *XENIX Reference Manual* for more information about setting terminal characteristics. Also, refer to the next section, "Stopping a Program."

2.8 Stopping a Program

You can abort the execution of most programs and commands by pressing the **INTERRUPT** key (perhaps called **DEL**, **DELETE**, **Ctrl-c**, or **RUBOUT** on your terminal). The **BREAK** key found on many terminals can also be used. Inside some programs, like most text editors, entering **INTERRUPT** stops whatever the program is doing without aborting the program itself. Throughout this manual, when we say "send an interrupt" we mean press the **INTERRUPT** key.

2.9 Logging Out

To end a session with XENIX, you must log out. This is done by entering **Ctrl-d** as the first character on a line. It is not sufficient just to turn off the terminal, since this does not log you out. Some programs can also be ended by entering **Ctrl-d**, so beware.

Chapter 3

Basic Concepts

3.1 Introduction

This chapter will give you an understanding of the basic concepts you need to function in the XENIX environment. After reading this chapter you should understand how the system's files, directories, and devices are organized and named, how commands are entered, and how a command's input and output can be manipulated. This chapter begins with a discussion of files.

3.2 Files

The file is the fundamental unit of the XENIX file system. In XENIX there are really three different types of files: ordinary files (what we usually mean when we say "file"), directories, and special files. Each of these types of files is described below.

3.2.1 Ordinary Files

Ordinary files typically contain textual information such as documents, data, or program sources. Executable binary files are also of this type. An ordinary file is simply a named concatenation of 8-bit bytes. Whether these bytes are interpreted as text characters, binary instructions, or program statements is up to the programs that examine them. Every ordinary file has the following attributes:

- A filename (not necessarily unique)
- A unique system number called an inode number
- A size in bytes
- A time of creation
- A time of modification
- A time of last access
- A set of access permissions

Files can be protected by assigning appropriate access permissions to assure privacy and security. This is done by providing read-write-execute permissions to files so that the user can control access by the owner, by a group of users, and by anyone else. By default, the owner of a file is its creator. The owner can read the file or write to it. By default, other users can read a file owned by another, but not write to it. File permissions can be altered with the **chmod** command. This command is discussed in Chapter 4 of this manual.

3.2.2 Special Files

Special files correspond to physical devices such as hard and floppy disks, lineprinters, terminals, and system memory. They are called "device special files". These files are not discussed in this manual.

3.2.3 Directory files

Directory files are read-only files containing information about the files or directories that are conceptually (but not physically) contained within them. This information consists of the name and inode number of each file or directory residing within the given directory. An inode number is a unique number associated with any given file. *All* files on the system have inode numbers. A name/inode number pair is called a link. The ls command is used to examine directory files and to list the information about the files conceptually within the named directory. With the inode number, the ls command can also find other information about a file.

The nesting of directories inside other directories is the way in which XENIX implements its characteristic tree-structured directory system. Directories are discussed further in the next section.

Like ordinary files, directories can be protected by assigning appropriate access permissions to assure privacy and security. This is done by giving read-write-search permissions to directories so that the user can control directory access by the owner, by a group of users, and by anyone else. Write permission determines whether files can be added or removed from a directory. By default, the owner of a directory is its creator, and the owner can read, create or remove files within that directory. Similarly by default, a user can read files within the directory of another, but not add or remove files. As with file permissions, directory permissions can be altered with the chmod command. Default permissions can be altered with the umask command.

3.2.4 Directory Structure

With multiple users and multiple projects, the number of files in a file system can proliferate rapidly. Fortunately, as mentioned earlier, XENIX organizes all files into a tree-structured directory hierarchy. This tree structure should be thought of as a physical world in which the user can move from place to place. "Places" are directories. Each user of the system has his own personal directory. Within that directory, the user may have directories or other subdirectories owned and controlled only by the user.

When you log in to XENIX, you are "in" your directory. Unless you take special action when you create a file, the new file is created in your working

directory. This file is unrelated to any other file of the same name in someone else's directory.

A diagram of part of a typical user directory is shown in Figure 3-1.

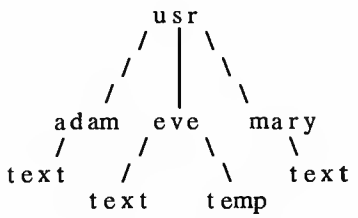

Figure 3-1 A Typical User Directory

In Figure 3-1, the *usr* directory contains each user's own personal directory. Notice that Mary's file named *text* is unrelated to Eve's. This is not important if all the files of interest are in Eve's directory, but if Eve and Mary work together, or if they work on separate but related projects, this division of files becomes handy indeed. For example, Mary could print Eve's text by typing:

```
pr /usr/eve/text
```

Similarly, Eve could find out what files Mary has by typing:

```
ls /usr/mary
```

3.3 File Systems

A file system is a set of files organized in a certain way. In XENIX, this set of files consists of all available resources including data files, directories, programs, lineprinters, and disks. Thus, the XENIX file system is a system for accessing all system resources.

To logically structure the resources of the system, the XENIX file system is organized hierarchically in an inverted "tree structure". See Figure 3-2 for an illustration of a typical tree-structured file system. In this typical tree of files, the root of the tree is at the top and branches of the tree grow downward. Directories correspond to nodes in the tree; ordinary files correspond to "leaves". If a directory contains a downward branch to other files or directories, then those files and directories are "contained" in the given directory. It is possible to name any file in the system by starting at the root (where the root is at the top) and traveling down any of the branches to the desired file. Similarly, you can specify any file in the

system, relative to any directory. Specification of these files depends on a knowledge of the XENIX naming conventions, discussed in the next section.

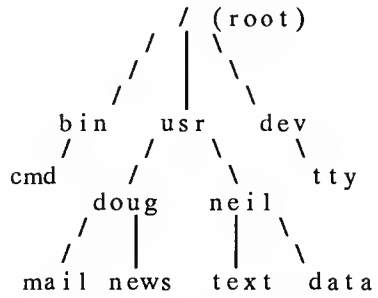

Figure 3-2 A Typical File System

In the typical tree-structured file system of Figure 3-2, the "tree" grows downward. The names *bin*, *usr*, *dev*, *doug*, and *neil* all represent directories, and are all nodes in the tree. In XENIX the name of the root directory is given the one-character name, "/". The names *mail*, *news*, *text*, and *data* all represent normal data files, and are all "leaves" of the tree. Note that the file *cmd* is the name of a command that can be executed. The name *tty* represents a terminal and is also represented in the tree.

3.4 Naming Conventions

Every single file, directory, and device in XENIX has both a filename and an absolute pathname. This pathname is a map of the file or directory's location in the system. The absolute pathname is unique to all names in the system; filenames are unique only within directories and need not be unique system-wide. This is similar to someone whose "global" name is John Albert Smith in a telephone directory, but who may be listed simply as John in an office phone list.

3.4.1 Filenames

A simple filename is a sequence of one to fourteen characters other than a slash (/). Every single file, directory, and device in the system has a filename. Filenames are used to uniquely identify directory contents. Thus, no two filenames in a directory may be the same. However, filenames in different directories may be identical.

Although you can use almost any character in a filename, it is best to confine filenames to the alphanumeric characters and the period. Other characters, especially control characters, are discouraged for use in filenames. When a filename contains an initial period, it is "hidden", and

is not displayed by the **lc** command. However the **ls–a** command will display the hidden files. The dash (–) is used in specifying command options, and should be avoided when naming files. In addition, the question mark (?), the asterisk (*), brackets ([and]), and all quotation marks should *never* be used in filenames, since they are treated specially when entering commands.

3.4.2 Pathnames

A pathname is a sequence of directory names followed by a simple filename, each separated from the previous name by a slash. If a pathname begins with a slash, it specifies a file that can be found by beginning a search at the *root* of the entire tree. Otherwise, files are found by beginning the search at the user's *current directory* (also known as the *working directory*). The current directory should be thought of as your location in the file system. Think of it as a physical place. When you change your current directory you are moving to some other directory or place in the file system.

A pathname beginning with a slash is called a *full* (or *absolute*) *pathname* because it does not vary with regard to the user's current directory. A pathname *not* beginning with a slash is called a *relative pathname*, because it specifies a path relative to the current directory. The user may change the current directory at any time by using the **cd** command. The user may display the current directory by using the **pwd** command.

3.4.3 Sample Names

Some sample names follow:

/	The absolute pathname of the root directory of the entire file system.
/bin	The directory containing most of the frequently used XENIX commands.
/usr	The directory containing each user's personal directory. The subdirectory, */usr/bin* contains frequently used XENIX commands not in */bin*.
/dev	The directory containing files corresponding to physical devices (e.g., terminals, lineprinters, and disks).
/dev/console	The name of the system master terminal.
/dev/tty	The name of the user's terminal.
/lib	The directory containing files used by some standard commands.

/tmp — This directory contains temporary scratch files.

/usr/joe/project/A
A typical full pathname; this one happens to be a file named *A* in the directory named *project* belonging to the user named *joe*.

bin/x — A relative pathname; it names the file *x* in subdirectory *bin* of the current working directory. If the current directory is /, it names */bin/x*. If the current directory is */usr/joe*, it names */usr/joe/bin/x*.

file1 — Name of an ordinary file in the current directory.

When using the XENIX system, each user resides "in" a directory called the current directory. All files and directories have a "parent" directory. This directory is the one immediately above, which "contains" the given file or directory. The XENIX file system provides special shorthand notations for this directory and for the current directory:

. The shorthand name of the current directory. Thus *./filexxx* names the same file as *filexxx*, if such a file exists in the current directory.

.. The shorthand name of the current directory's parent directory. The shorthand name ../.. refers to the directory that is two levels "above" the current directory

3.4.4 Special Characters

XENIX provides a pattern-matching facility for specifying sets of filenames that match particular patterns. For example, examine the problem that occurs when naming the parts of a large document, such as a book. Logically, it can be divided into many small pieces such as chapters or sections. Physically, it must be divided too, since the XENIX editor vi cannot handle really big files.

Thus, you should divide a large document into several files. The points at which the document is divided should follow a logical order. You might have a separate file for each chapter:

```
chap1
chap2
...
```

Or, if each chapter is broken into several files, you might have:

```
chap1.1
chap1.2
chap1.3
...
chap2.1
chap2.2
...
```

You can then tell at a glance where a particular file fits into the whole.

There are other advantages to a systematic naming convention that are not so obvious. What if you want to print the whole book on the lineprinter? You could enter:

```
lpr chap1.1 chap1.2 chap1.3...
```

but you will tire of this quickly and will probably even make mistakes. Fortunately, there is a shortcut: a sequence of names containing a common pattern can be specified with the use of special characters. The special characters discussed in this chapter are:

- * Matches zero or more characters
- [] Matches any character inside the brackets
- ? Matches any single character

For example, you can enter:

```
lpr chap*
```

The asterisk (*), sometimes called "star" or "splat" in XENIX, means "zero or more characters of any type", so this translates into "send all files whose names begin with the word "chap" to the lineprinter".

This shorthand notation is not a unique property of the **lpr** command; it can be used in any command.

Using this fact, you can list the names of the files in the book by typing:

```
ls chap*
```

This produces

```
chap1.1
chap1.2
chap1.3
...
```

The star is not limited to the last position in a filename; it can be used anywhere and can occur several times. A star by itself matches all filenames not containing slashes or beginning with periods, so:

```
cat *
```

displays all files in the current directory on your terminal screen.

The star is not the only pattern-matching feature available. Suppose you want to print only chapters 1 through 4, and 9. You can say:

```
lpr chap[12349]*
```

The brackets ([and]) mean "match any of the characters inside the brackets." A range of consecutive letters or digits can be abbreviated, so you can also do this with:

```
lpr chap[1-49]*
```

(Note that this does *not* match forty-nine filenames, but only five.) Letters can also be used within brackets: "[a-z]" matches any character in the range "a" through "z".

The question mark (?) matches any single character, so

```
ls ?
```

lists all files that have single-character names, and

```
ls -l chap?.1
```

lists information about the first file of each chapter (i.e., *chap1.1*, *chap2.1*, ...).

If you need to turn off the special meaning of any of the special characters (*, ?, and [...]) enclose the entire argument in single quotation marks.

For example, the following command will print out only files named "?" rather than all one-character filenames:

```
ls '?'
```

Pattern-matching features are discussed further in Chapter 4 of the XENIX *User's Guide*, "The Shell."

3.5 Commands

Commands are used to invoke executable programs. When you enter the name of a command, XENIX reads the command line that you have entered, looks for a program with the given name, and then executes the program if it finds it. Command lines may also contain arguments that specify options or files that the program may need. The command line and command syntax are discussed in the next two sections.

3.5.1 Command Line

Whether you are entering commands at a terminal, or XENIX is reading commands from a file, XENIX always reads commands from command lines. The command line is a line of characters that is read by the shell command interpreter to determine what actions to perform. This interpreter, or "shell" as it is known, reads the names of commands from the command line, finds the executable program corresponding to the name of the command, then executes that program. When the program finishes executing, the shell resumes reading the command line. Thus, when you are entering at a terminal, you are editing a line of text called the *command-line buffer* that becomes a command line only when you press **RETURN**. This command-line buffer can be edited with the **BKSP** and Ctrl-u keys. Pressing **RETURN** causes the command-line buffer to be submitted to the shell as a command line. The shell reads the command line and executes the appropriate command. If you press **INTERRUPT** before you press **RETURN**, the command-line buffer is erased. Multiple commands can be entered on a single command line provided they are separated by a semicolon (;). For example, the following command line prints out the current date and the name of the current working directory:

```
date ; pwd
```

Commands can be submitted for processing in "the background" by appending an ampersand (&) to the command line. This mode of execution is similar to "batch" processing on other systems. The main advantage

to placing commands in the background is that you can execute other commands from your terminal in the "foreground" while the background commands execute. Thus:

```
du /usr > diskuse&
```

determines the disk usage in the directory */usr*, a fairly time-consuming operation, without tying up your terminal. Note that the output is placed in the file *diskuse* by redirecting output with the greater-than symbol. Redirection is discussed in Section 3.6.1.

3.5.2 Syntax

The general syntax for commands is as follows:

cmd [*switches*][*arguments*] [*filename*] [...]

By convention, command names are lowercase. Switches, also called options, are flags that select various options available when executing the command. They are optional and usually *precede* other arguments and filenames. Switches consist of a dash prefix (-) and an identifying letter. For example, the ls command's **-l** switch (pronounced "minus ell") specifies a long directory listing and the command

```
ls -r
```

specifies a directory listing in reverse alphabetical order. In some cases, switches can be grouped to form a single switch argument. For example, the command

```
ls -rl
```

is really a combination of two switches, where the **-rl** switch selects the option that lists all files in the directory in both reverse alphabetical order and with the long format.

Sometimes multiple switches must be given separately, as in:

```
copy -a -v source destination
```

Here the **-a** switch tells the **copy** command to ask the user for confirmation before copying the *source* to the *destination.* The **-v** switch specifies the "verbose" option, which reports copying as it happens.

Other arguments, such as search strings, can also be given, as in:

```
grep ´ string of text ´ outfile
```

In the above example,

```
´string of text´
```

is a single argument and is the search string the **grep** command searches for in the file *outfile*. *filename* is the argument that specifies the name of a file required by the command.

Most commands are executable programs compiled by the C compiler or by some other language compiler. Some commands are executable command files called "shell procedures". Shell procedures are discussed in Chapter 4 of the XENIX *User's Guide*, "The Shell."

3.6 Input and Output

By default, XENIX assumes that terminal input comes from the terminal keyboard and output goes to the terminal screen. To illustrate typical command input and output, enter:

```
cat
```

This command now expects input from your keyboard. As input, it accepts as many lines of text as you enter until you press Ctrl-d as an end-of-file or end-of-transmission indicator.

For example, enter:

```
this is two linesRETURN
of inputRETURN
Ctrl-d
```

When you press Ctrl-d, input ends. The **cat** command immediately outputs each line as you enter it. Since output is sent to the terminal screen by default, that is where the two lines are sent. Thus, the complete session will look like this on your terminal screen:

```
$ cat
this is two lines
this is two lines
of input
of input
$
```

The flow of command input and output can be "redirected" so that input comes from a file instead of from the terminal keyboard and output goes to a file or lineprinter, instead of to the terminal screen. In addition, you can create "pipes" to make the output from one command become the input to another. Redirection and pipes are discussed in the next two subsections. When you use **cat** to send input to a file or pipe, the output is not sent until the Ctrl-d end-of-transmission indicator is entered.

3.6.1 Redirection

In XENIX a file can replace the terminal for either input or output. For example:

```
ls
```

displays a list of files on your terminal screen. But if you say:

```
ls > filelist
```

a list of your files is placed in the file *filelist* (which is created if it does not exist). The symbol for output redirection, the greater-than sign (>), means "put the output from the command into the following file, rather than display it on the terminal screen." As another example of output redirection, you can combine several files into one by capturing the output of **cat** in a file:

```
cat f1 f2 f3 > temp
```

The output append symbol (>>) operates very much like the output redirection symbol, except that it means "add to the end of". So:

```
cat file1 file2 file3 >> temp
```

means "concatenate *file1*, *file2*, and *file3* to the end of whatever is already in *temp*, instead of overwriting and destroying the existing contents." As with normal output redirection, if *temp* doesn't exist, it is created for you.

In a similar way, the input redirection symbol (<) means "take the input for a program from the following file, instead of from the terminal". Thus, you could make a script of editing commands and put them into a file called *script*. Then you could execute the commands in the script on a file by typing:

```
ed file < script
```

As another example, if you used an editor to prepare a letter in the file *letter.txt*, you could send it to several people with:

```
mail adam eve mary joe < letter.txt
```

3.6.2 Pipes

One of the major innovations of the XENIX system is the concept of a "pipe". A pipe is simply a way to connect the output of one command to the input of another, so that the two run as a sequence of commands called a pipeline.

For example:

```
sort frank.txt george.txt hank.txt
```

combines the three files named *frank.txt*, *george.txt*, and *hank.txt*, then sorts the output. Suppose that you want to then find all unique words in these files and view the result. You could enter:

```
sort frank.txt george.txt hank.txt > temp1
uniq < temp1 > temp2
more temp2
rm temp1 temp2
```

But this is more work than is necessary. What you want is to take the output of **sort** and connect it to the input of **uniq**, then take the output of uniq and connect it to **more.** You would use the following pipe:

```
sort frank.txt george.txt hank.txt | uniq | more
```

The vertical bar character (|) is used between the **sort** and **uniq** commands to indicate that the output from **sort**, which would normally have been sent to the terminal, is to be redirected from the terminal to the standard input of the **uniq** command, which in turn sends its output to the **more** command for viewing.

There are many other examples of pipes. For example, this command formats and paginates a list of your files in three columns:

```
ls | pr -3
```

The program **wc** counts the number of lines, words, and characters in its input, and **who** prints a list of users currently logged on, one per line. Thus, this command tells you the number of users who are logged in by counting the number of lines that comes from the **who** command:

```
who | wc -l
```

This command counts the number of files in the current directory:

```
ls | wc -l
```

Notice the difference in output between **wc -l** and **wc**. By default, **wc** tells you how many lines, word and characters, there are in the input. However, **wc -l** tells you only how many lines.

Any program that reads from the terminal keyboard can read from a pipe instead. Any program that displays output to the terminal screen can send input to a pipe. You can have as many elements in a pipeline as you wish.

Chapter 4

Tasks

4.1 Introduction

This chapter explains how to perform common tasks on XENIX. The individual commands used to perform these tasks are discussed more thoroughly in the XENIX *Reference Manual.*

4.2 Gaining Access to the System

To use the XENIX system, you must first gain access to it by logging in. When you log in you are placed in your own personal working area. Logging in, changing your password, and logging out are described below.

4.2.1 Logging In

Before you can log in to the system, you must be given a system "account." Your name must be added to the user list, and you must be given a password and a mailbox.

Depending on how your system is administered, you may have to add your name to the user list yourself, or someone else may be assigned this task. If you must add your own account to the system, see the XENIX *Operations Guide* and **mkuser**(C) in the XENIX *Reference Manual* for more information. This section assumes your account has already been set up.

Normally, the system sits idle and the prompt "login:" appears on the terminal screen. If your screen is blank, or displays nonsense characters, press the **INTERRUPT** key a few times.

When the "login:" prompt appears, follow these steps:

1. Enter your login name and press **RETURN**. If you make a mistake, press **Ctrl-u** to start the line again. After you press **RETURN** the word "Password:" appears on your screen.

2. Enter your password carefully, then press **RETURN**. The letters do not appear on your screen as you enter, and the cursor does not move. If you make a mistake, press **RETURN** to restart the login procedure.

If you have entered your login name and password correctly the "prompt character" appears on the screen. This is usually a dollar sign($). The prompt tells you that the XENIX system is ready to acccept commands from the keyboard.

If you make a mistake, the system displays the message:

```
Login incorrect
login:
```

If you get this message, follow the above procedure again. You must enter all the letters of your user name and password correctly before you are given access to the system; XENIX does not allow you to correct your mistakes when entering your password.

Depending on how your system is set up, after you log in you may see a "message of the day" that says something like "Welcome to XENIX", or an announcement that is of interest to all users.

4.2.2 Logging Out

The logout procedure is simple–all you need to do is press:

```
Ctrl-d
```

alone on a line. In general, **Ctrl-d** signifies the end-of-file in XENIX, and is often used within programs to signal the end of input from the keyboard. In such cases, **Ctrl-d** will *not* log you out; it will simply terminate input to a particular program if you are within that program. This means that it may sometimes be necessary to press **Ctrl-d** several times before you can log yourself out. For example, if you are in the **mail** program you must press **Ctrl-d** once to exit the mail program, then again to log out.

4.2.3 Changing Your Password

To prevent unauthorized users from gaining access to the system, each authorized user must have a password. When you are first given an account on a XENIX system you are assigned a password by the system administrator. Some XENIX systems require you to change your password at regular intervals. Whether yours does or not, it is a good idea to change your password regularly to maintain system security. This section tells you how to change your password.

Use the **passwd** command to change your password. Follow these steps:

1. Enter:

   ```
   passwd
   ```

   and press **RETURN**.

The following message appears:

```
Changing password for user
Old password:
```

2. Carefully enter your old password. It is not displayed on the screen. If you make a mistake, press **RETURN**. The message "Sorry" appears, then the system prompt. Begin again with step 1.

3. When you have entered your old password the message:

```
New password:
```

appears. Enter your new password and press **RETURN**.

4. The message:

```
Re-enter new password:
```

appears. Enter your new password again. If you make a mistake, press **RETURN**. The message:

```
They don't match; try again
```

appears, and you must begin again with step 1. When you have completed the procedure, the system prompt appears.

4.3 Configuring Your Terminal

On most systems, the standard console terminal is already configured for use with XENIX. However, other terminals of various types may be connected to a XENIX system. In these cases it is important to know how to set terminal options and how to specify the terminal you are using. You may also want to change the standard configuration of the standard console terminal. The following section discusses these topics.

4.3.1 Changing Terminals

The terminal type is displayed each time you log in. If you ever need to log in to XENIX on a terminal of a type different than the terminal you normally use, you may need to change your environment by editing the *.profile* file in your home directory. To do this, use a text editor to locate the **tset** line that looks something like this:

```
eval 'tset -m :\?unknown -s -r -Q'
```

Change *unknown* in this line to the terminal type of your terminal. For example, if you normally log in on a vt100 terminal, change the line to:

```
eval 'tset -m :\?vt100 -s -r -Q'
```

Each time you log in you then see the message:

```
TERM = (vt100)
```

Press **RETURN** and the terminal type is set to vt100, or enter another terminal type and press **RETURN**.

4.3.2 Setting Terminal Options

There are a number of terminal options that can be set with the command **stty**. When entered without parameters, **stty** displays the current terminal settings. For example, typical output might look like this:

```
speed 9600 baud
erase '^h' ; kill '^u'
even -nl
```

Each of the above characteristics can be set with **stty**. For more information, see **stty**(C) in the XENIX *Reference Manual*.

4.4 Editing the Command Line

When you sit in front of a terminal and enter commands at your keyboard, there are a number of special keys that you can use. The most useful ones are described below.

4.4.1 Entering a Command Line

From your terminal, entering a command line consists of typing characters then pressing **RETURN**. Once you have pressed **RETURN** the computer reads the command line and commands specified on that line are executed. You may enter as many command lines as you want without waiting for commands to complete, because XENIX supports type-ahead of characters.

4.4.2 Erasing a Command Line

When entering commands, typing errors are bound to occur. To erase the current command line, press **Ctrl-u**.

4.4.3 Halting Screen Output

In many cases, you will be examining the contents of a file on the terminal screen. For longer files, the contents will often scroll off the screen faster than you can examine them. To temporarily halt a program's output to the terminal screen, press **Ctrl-s**. To resume output, press **Ctrl-q**.

4.5 Manipulating Files

File manipulation (creating, displaying, combining, copying, moving, naming, and deleting files), is one of the most important capabilities an operating system provides. The XENIX commands that perform these functions are described in the following sections.

4.5.1 Creating a File

To create a file and place text in it, use the editor **vi**, described in Chapter 2 of the XENIX *User's Guide*, "vi: A Text Editor." If for some reason you wish to create an empty file, enter:

```
> filename
```

where *filename* is the name of the empty file. In general, new files are created by commands as needed.

4.5.2 Displaying File Contents

The **more** command displays the contents of a file one screenful at a time. It has the form:

```
more options filename
```

more is useful for looking at a file when you do not want to make changes to it. For example, to display the contents of the file *memos*, enter:

```
more memos
```

more can be invoked with options that control where the display begins, and how the file is displayed.

These options include:

+linenumber
: Begins the display at the line in the file designated by *linenumber*.

+/*text*
: Begins the display two lines before *text*, where *text* is a word or number. If *text* is two or more words, they must be enclosed in double quotation marks.

-c
: Redraws the screen instead of scrolling.

-r
: Displays control characters, which are normally ignored by **more**.

To begin looking at the file *memo* at the first occurrence of the words "net gain", for example, enter:

```
more +/"net gain" memo
```

If the file is more than one screenful long, the percentage of the file that remains is displayed on the bottom line of the screen. To look at more of the file, use the following scrolling commands:

RETURN
: Scrolls down one line.

d
: Scrolls down one-half screen.

SPACE
: Scrolls down a full screen.

n **SPACE**
: Scrolls down *n* lines.

.
: Repeats the previous command.

You cannot scroll backward, toward the beginning of the file.

You can search forward for patterns in **more** with the slash (/) command.

For example, to search for the pattern "net gain", enter:

```
/net gain/
```

and press **RETURN**. **more** displays the message:

at the top of the screen, and scrolls to a location two lines above "net gain."

If you are looking at a file with **more** and decide you want to change the file, you can invoke the **vi** editor by pressing:

```
v
```

See Chapter 2 of the XENIX *User's Guide*, "vi: A Text Editor," for information on using **vi**.

more quits automatically when it reaches the end of a file. To exit **more** before the end of a file, enter:

```
q
```

The **head** and **tail** commands display the first and last ten lines of a file, respectively. They are useful for checking the contents of a particular file.

For example, to look at the first ten lines of the file *memo,* enter:

```
head memo
```

You can also specify how many lines the **head** and **tail** commands display. For example:

```
tail -4 memo
```

displays the last four lines of *memo.*

The **cat** command also displays the contents of a file. **cat** scrolls the file until you press **Ctrl-s** to stop it. Pressing **Ctrl-q** will continue the scrolling. **cat** stops automatically at the end of a file. If you wish to stop the display before the end of the file, press **INTERRUPT**. To display the contents of one file, enter:

```
cat file1
```

To display the contents of more than one file, enter:

```
cat file1 file2 file3
```

4.5.3 Combining Files

The **cat** command is frequently used to combine files into some other new file.

Thus, to combine the two files named *file1* and *file2,* into a new file named *bigfile,* enter:

```
cat file1 file2 >bigfile
```

Note here that we are putting the contents of the two files into a new file with the name *bigfile.* The greater than sign (>) is used to *redirect* output of the **cat** command to the new file.

You can also use **cat** to append one file to the end of another file. For example, to append *file 1* to *file 2*, enter:

```
cat file1 >> file2
```

The contents of *file 1* are added to *file 2*. *file 1* still exists as a separate entity.

4.5.4 Moving a File

The **mv** command moves a file into another file in the same directory, or into another directory.

For instance, to move a file named *text* to a new file named *book*, enter:

```
mv text book
```

After this move is completed, no file named *text* will exist in the working directory, because the file has been renamed *book*.

To move a file into another directory, give the name of the destination directory as the final name in the **mv** command. For instance, to move *file1* and *file2* into the directory named */tmp*, enter:

```
mv file1 file2 / tmp
```

The two files you have moved no longer exist in your working directory, but now exist in the directory */tmp*. The above command has exactly the same effect as entering the following two commands:

```
mv file1 /tmp/file1
mv file2 /tmp/file2
```

The **mv** command always checks to see if the last argument is the name of a directory and, if so, all files designated by filename arguments are moved into that directory.

4.5.5 Renaming a File

To rename a file, simply "move" it to a file with the new name: the old name of the file is removed. Thus, to rename the file *anon* to *johndoe*, enter:

```
mv anon johndoe
```

Note that moving and renaming a file are essentially identical operations.

4.5.6 Copying a File

There are two forms of the **cp** command: one in which files are copied into a directory, and another in which a file is copied to another file. Thus, to copy three files into a directory named *filer,* enter:

```
cp file1 file2 file3 filer
```

In the above command, three files are copied into the directory *filer;* the original versions still reside in the working directory. Note that the filenames are identical in the two directories. Like the **mv** command, **cp** always checks to see if the last argument is the name of a directory, and, if so, all files designated by filename arguments are copied into that directory.

To create two copies of a file in your own working directory, you must rename the copy. To do this, the copy command can be invoked as follows:

```
cp file filecopy
```

After the above command has executed, two files with identical contents reside in the working directory. To learn how to copy directories, see section 4.6.7, "Copying a Directory", later in this chapter.

4.5.7 Deleting a File

To delete or remove files, enter:

```
rm file1 file2
```

In the above command, the files *file1* and *file2* are removed from your working directory.

The command:

```
rm -i file1 file2
```

allows you to interactively remove files by asking you if you really want to delete each of the files *file1* and *file2.* If you press *y* followed by a **RETURN**, the given file is removed; if you press *n* the file is left untouched. This command is useful when cleaning up a directory that contains many files.

4.5.8 Finding Files

The **find** command searches for files that have a specified name. **find** is useful for locating files that have the same name, or just for finding a file if you have forgotten which directory it is in.

The command has the form:

find *pathname* **-name** *filename* **-print**

The *pathname* is the pathname of the directory you want to search. **find** searches recursively, that is, it starts at the named directory and searches downward through all files and subdirectories under the directory specified in *pathname*.

The **-name** option indicates that you are searching for files that have a specific *filename*. (There are other search conditions you can use with **find**; see **find**(C) in the XENIX *Reference Manual*.)

filename is the name of the file you are searching for.

The **-print** option indicates you want to print the pathnames of all the files that match *filename* on your terminal screen. You may direct this output to a file instead of your screen with the output redirection symbol, >. (There are other actions that can be performed with **find**, such as removing and moving files; see **find**(C) in the XENIX *Reference Manual*.) For example, the following command finds every file named *memo* in the directory */usr/joe* and all its subdirectories:

```
find /usr/joe -name memo -print
```

The output might look like this:

```
/usr/joe/memo
/usr/joe/accounts/memo
/usr/joe/meetings/memo
/usr/joe/mail/memo
```

4.5.9 Linking a File to Another File

The **ln** command joins two files in different directories so that when the file is changed in one directory, it is also changed in the other directory. This can be useful if several users need to share information, or if you want a file to appear in more than one directory. This command has the form:

ln *file newfile*

where *file* is the original file, and *newfile* is the new, linked file. For example, the following command links *memos* in */usr/joe* to *joememos* in */usr/mary:*

```
ln /usr/joe/memos /usr/mary/joememos
```

Whenever */usr/joe/memos* is updated, the file */usr/mary/joememos* is also changed.

When you link files a name is associated with an *inode.* An inode specifies a unique set of data on the disk. One or more names can be associated with this data. Thus, the above command assures that the files *dir1/file1* and *dir2/file2* have identical contents.

There are three things that are not immediately obvious about linking files:

1. Linking large sets of files to other parallel files can save a considerable amount of disk space.

2. Linking files used by more than one person is risky, because any party can alter the file and thus affect the contents of all files linked to it.

3. Removing a file from a directory does not remove other links to the file. Thus the file is not truly deleted from the system. For example, if you delete a file that has 4 links, 3 links remain.

For more information about linking see **ln**(C) in the XENIX *Reference Manual.*

4.6 Manipulating Directories

Because of the hierarchical organization of the file system, there are many directories and subdirectories in the XENIX system. Within the file system are directories for each user of the system. Within your user directory you can create, delete, and copy directories. Commands that let you manipulate directories are described in the following sections.

4.6.1 Printing the Name of Your Working Directory

All commands are executed relative to a "working" directory. The name of this directory is given by the **pwd** command, which stands for "print working directory." For instance, if your current working directory is */usr/joe*, when you enter:

```
pwd
```

you will get the output:

```
/usr/joe
```

You should always think of yourself as residing "in" your working directory.

4.6.2 Listing Directory Contents

You can list the contents of a directory with the **lc** command. This command sorts the names of files and directories in a given directory, and lists them in columns. If no directory name is given, **lc** lists the contents of the current directory. The **lc** command has the form:

```
lc options name
```

For example, to list the contents of the directory *work*, enter:

```
lc work
```

Your output might look like this:

```
accounts    meetings    notes
mail        memos       todo
```

If no *name* is specified, **lc** lists the contents of the current directory. If *accounts* is the current directory, for example, the command:

```
lc
```

lists the names of the files and subdirectories in that directory.

The following options control the sort order and the information displayed by the **lc** command:

- **–a** Lists all files in the directory, including the "hidden" files (filenames that begin with a dot, such as *.profile* and *.mailrc*).
- **–r** Lists names in reverse alphabetical order.
- **–t** Lists names in order of last modification, the latest (most recently modified) first. When used with the **–r** option, lists the oldest first.

-R Lists all files and directories in the current directory, plus each file and directory *below* the current one. The "R" stands for "recursive."

-F Marks directories with a backslash(\) and executable files with an asterisk (*).

The **ls** command works much like the **lc** command except that it lists files in vertical, rather than collumnar, form. The **ls -l** command gives a "long" listing of a directory, producing an output that might look something like this:

```
total 501
drwxr-x---  2 boris grp1 272  Apr 5 14:33 dir1
drwxr-x---  2 enid  grp1 272  Apr 5 14:33 dir2
drwxr-x---  2 iris  grp1 592  Apr 6 11:12 dir3
-rw-r-----  1 olaf  grp2 282  Apr 7 15:11 file1
-rw-r-----  1 olaf  grp2 72   Apr 7 13:50 file2
-rw-r-----  1 olaf  grp2 1403 Apr 1 13:22 file3
```

Reading from left to right, the information given for each file or directory includes:

- Permissions
- Number of links
- Owner
- Group
- Size in bytes
- Time of last modification
- Filename

The information in this listing and how to change permissions are discussed below in Section 4.8, "Using File and Directory Permissions."

For more information about listing the contents of a directory, see ls(C) in the XENIX *Reference Manual.*

4.6.3 Creating a Directory

To create a subdirectory in your working directory, use the **mkdir** command. For instance, to create a new directory named *phonenumbers*, simply enter:

```
mkdir phonenumbers
```

After this command has been executed, a new empty directory will exist in your working directory.

4.6.4 Removing a Directory

To remove a directory located in your working directory, use the **rmdir** command. For instance, to remove the directory named *phonenumbers* from the current directory, simply enter:

```
rmdir phonenumbers
```

Note that the directory *phonenumbers* must be *empty* before it can be removed; this prevents catastrophic deletions of files and directories. If you want to live dangerously, it is possible to recursively remove the contents of a directory using the **rm** command, but that will not be explained here. See **rm**(C) in the XENIX *Reference Manual* for more information.

4.6.5 Renaming a Directory

To rename a directory, use the **mv** command. For instance, to rename the directory *little.dir* to *big.dir*, enter:

```
mv little.dir big.dir
```

This is a simple renaming operation; no files are moved.

4.6.6 Moving a Directory

The **mv** command also moves directories. This command has the form:

```
mv oldirectory newdirectory
```

where *newdirectory* is a directory that already exists.

For example, to move the directory */usr/joe/accounts* into */usr/joe/overdue* enter:

```
mv /usr/joe/accounts /usr/joe/overdue
```

The new pathname of */usr/joe/accounts* is */usr/joe/overdue/accounts.*

4.6.7 Copying a Directory

The **copy** command copies directories. This command has the form:

```
copy options olddir newdir
```

To copy all the files in the directory */usr/joe/memos* into */usr/joe/notes* enter:

```
copy /usr/joe/memos /usr/joe/notes
```

The files in */usr/joe/memos* are copied into */usr/joe/notes*. The **copy** command has the following options:

- -l Links the copied files to the original.
- -m Gives the copied files the same modification dates as the original files.
- -r Copies the directory recursively, i.e., copies all the directories under the named directory.

4.7 Moving in the File System

When using the XENIX system, it helps to imagine a large tree structure of files and directories. Each directory should be thought of as a place that you can move into or out of. At all times you are "someplace" in the tree structure. This place is called either your working directory or current directory. The commands used to find out where you are and to move around in the tree structure are discussed below.

4.7.1 Finding Out Where You Are

Your current location in the file system is the name of the working directory. You can find out this name by using the **pwd** command, which stands for "print working directory." For example, if you are in the directory */usr* then entering the command:

```
pwd
```

prints out the name:

```
/usr
```

4.7.2 Changing Your Working Directory

Your working directory represents your location in the file system: it is "where you are" in XENIX. To alter this location in the XENIX file system, use the change directory (**cd**) command:

```
cd
```

This changes your working directory to your home directory. To move to any other directory, specify that directory as an argument to **cd**.

For instance, the following command:

```
cd /usr
```

moves you to the */usr* directory. Because you are always "in" your working directory, changing working directories is much like "traveling" from directory to directory.

To move up one directory from your current directory, enter:

```
cd ..
```

For example, the above command would move you from the directory */usr/joe/work* to */usr/joe*. Similarly, the command:

```
cd ../..
```

would move you from the directory */usr/joe/work* to */usr*, moving you up *two* directories.

4.8 Using File and Directory Permissions

The XENIX system allows the file owner to restrict access to files and directories, limiting who can read, write and execute files owned by him. To determine the permissions associated with a given file or directory, use the **ls -l** command. The output from the **ls -l** command should look something like this:

```
total 501
drwxr-x---  2 boris grp1 272  Apr 5 14:33 dir1
drwxr-x---  2 enid  grp1 272  Apr 5 14:33 dir2
drwxr-x---  2 iris  grp1 592  Apr 6 11:12 dir3
-rw-r-----  1 olaf  grp2 282  Apr 7 15:11 file1
-rw-r-----  1 olaf  grp2 72   Apr 7 13:50 file2
-rw-r-----  1 olaf  grp2 1403 Apr 1 13:22 file3
```

Permissions are indicated by the first ten characters of the output. The permissions for dir1, the first file in the above list, are:

```
drwxr-x---
```

The first character indicates the type of file and must be one of the following:

- **-** Indicates an ordinary file.
- **d** Indicates a directory.
- **c** Indicates a character special device such as a lineprinter or terminal.
- **b** Indicates a block special device such as a hard or floppy disk.
- **n** Indicates a name special file (i.e., a semaphore used for controlling access to some resource).
- **s** Indicates a shared data file.
- **p** Indicates a named pipe.

From left to right, the next nine characters are interpreted as three sets of three permissions each. Each respective set of three indicates the following permissions:

- Owner permissions
- Group permissions
- All other user permissions

Within each set, the three characters indicate permission to read, to write, and to execute the file as a command, respectively. For a directory, "execute" permission means permission to search the directory for any included files or directories.

Ordinary file permissions have the following meanings:

r	The file is readable.
w	The file is writeable.
x	The file is executable.
-	The indicated permission is not granted.

For directories, permissions have the following meanings:

r	Files can be listed in the directory; the directory must also have "x" permission.
w	Files can be created or deleted in the directory; as with "r", the directory itself must also have "x" permission.
x	The directory can be searched. A directory must have "x" permission before you can move to it with the **cd** command (i.e., **cd** to it), access a file within it, or list the files in it. Remember that a user must have "x" permission to do anything useful to the directory.

The following are some typical directory permission combinations:

d---------	No access at all. This is the mode that denies access to the directory to a class of users.
drwx------	Allows access by only the owner to use **lc**, create files, delete files, access files (subject to file permissions), and **cd** to the directory. This is the typical permission for the owner of a directory.
drwxr-x---	Allows access by members of the group to use **lc**, and access files subject to file permissions. Group members can **cd** to this directory, but cannot create or delete files in it. This is the typical permission an owner gives to others who need access to files in his directory.
drwx--x--x	With these permission settings users other than the owner cannot use **lc** but can **cd** to the directory. Other

users can only access a file within this directory by its exact name; they cannot use special characters. Files cannot be created or deleted in the directory by anyone except the owner. This mode is rarely used, but can be useful if you want to give someone access to a specific file in a directory without permitting access to other files in the same directory.

This chapter discusses ordinary files, executable files, and directories only. For information about other types of files, see ls (C) in the XENIX *Reference Manual*.

4.8.1 Changing Permissions

The **chmod** command changes the read, write, execute, and search permissions of a file or directory. This command is useful if you have created a file in one mode, but want to give others permission to read, write or execute it.

The **chmod** command has the form:

chmod *instruction filename*

The *instruction* segment of the command indicates which permissions you want to change for which class of users. There are three classes of users, and they are are indicated as follows:

u User, the owner of the file or directory

g Group, the group the owner of the file belongs to

o Other, all users of the system

a All classes of users

There are three types of permissions, as follows:

r Read, which allows permitted users to look at but not change or delete the file.

w Write, which allows permitted users to change or even delete the file.

x Execute, which allows permitted users to execute the file as a command.

For example, assume *file1* exists with the following permissions:

```
-rw-r-----
```

In the above example, the owner of the file has read and write permission, group members have read permission, and others have no access at all.

To give *file1* read permission for *all* classes of users, enter:

```
chmod a+r file1
```

In the instruction segment of the command (a+r) the "a" stands for "all."

The resulting permissions are:

```
-rw-r--r--
```

For *file1* with the attributes:

```
-rw-------
```

The following command gives write and execute permissions to members of a group only:

```
chmod g+wx file1
```

This command would alter the permission attributes so they look like this:

```
-rw--wx---
```

To remove write and execute permission by the user (owner) and group associated with *file1*, enter:

```
chmod ug-wx file1
```

4.8.2 Changing Directory Search Permissions

Directories also have an execute permission. This attribute signifies search permission, rather than execute permission, since directories cannot be executed. If this permission is denied to a particular user, then that user cannot even list the names of the files in the directory.

For example, assume that the directory *dir1* has the following attributes:

```
drwxr-xr-x
```

To remove search permission for other users to examine *dir1*, enter:

```
chmod o-xr dir1
```

The new attributes for *dir1* are:

```
drwxr-x---
```

4.9 Processing Information

In many cases, files will contain information that you may want to process. Various utility programs exist on XENIX to process information. A set of these programs and their uses are described in the following sections.

4.9.1 Comparing Files

To compare two text files use the **diff** command to print out those lines that differ between the files that you specify.

For example, suppose that a file named *men* has the contents:

```
Now is the time for all good men to
Come to the aid of their party.
```

and that a file named *women* has the following contents:

```
Now is the time for all good women to
Come to the aid of their party.
```

If this is the case, then the command:

```
diff men women
```

produces the following results:

```
1c 1
< Now is the time for all good men to
---
> Now is the time for all good women to
```

A three-way difference listing can be created with the **diff3** command. For information about **diff3** see **diff3**(C) in the XENIX *Reference Manual.*

4.9.2 Echoing Arguments

The **echo** command echos arguments to the standard output. For example, entering:

```
echo hello
```

outputs:

```
hello
```

on the terminal screen. To output several lines of text, surround the echoed argument in double quotation marks and press **RETURN** between

lines. A secondary prompt will appear until you enter the final double quotation mark. For example, enter:

```
echo "Now is the time
For all good men
To come to the
Aid of their party."
```

This will output:

```
Now is the time
For all good men
To come to the
Aid of their party.
```

echo is particularly useful if you should ever program in the shell command language. For more information about the shell, see Chapter 4, "The Shell", XENIX *User's Guide*.

4.9.3 Sorting a File

One of the most useful file processing commands is **sort**. By default, **sort** sorts the lines of a file according to the ASCII collating sequence (i.e., it alphabetizes them).

For example, to sort a file named *phonelist*, enter:

```
sort phonelist
```

In the above case, the sorted contents of the file are displayed on the screen. To create a sorted version of *phonelist* named *phonesort*, enter:

```
sort phonelist >phonesort
```

Note that **sort** is useful for sorting the output from other commands. For example, to sort the output from execution of a **who** command, enter:

```
who | sort >whosort
```

This command takes the output from **who**, sorts it, and then sends the sorted output to the file *whosort*.

A wide variety of options are available for **sort**. For more information, see **sort**(C) in the XENIX *Reference Manual*.

4.9.4 Searching for a Pattern in a File

The **grep** command selects and extracts lines from a file, printing only those lines that match a given pattern. For example, to print out all lines in a file containing the word "tty38", enter:

```
grep 'tty38' file
```

In general, you should always enclose the pattern you are searching for in single quotation marks ('), so that special metacharacters are not expanded unexpectedly by the shell.

As another example, assume that you have a file named *phonelist* that contains a name followed by a phone number on each line. Assume also that there are several thousand lines in this list. You can use **grep** to find the phone number of someone named Joe, whose phone number prefix is 822, as follows:

```
grep 'joe' phonelist | grep '822-' >joes.number
```

grep finds all occurrences of lines containing the word "joe" in the file *phonelist*. The output from this command is then filtered through another **grep** command, which searches for an "822-" prefix, thus removing any unwanted joes. Finally, assuming that a unique phone number for joe exists with the "822-" prefix, that name and number are placed in the file *joes.number*.

For more information about **grep**, its relatives **fgrep** and **egrep**, and the types of patterns it can be used to search for (called regular expressions) see **grep**(C) in the XENIX *Reference Manual*.

4.9.5 Counting Words, Lines, and Characters

wc is a utility for counting words in a file. The letters "wc" stand for word count. Words are presumed to be separated by punctuation, spaces, tabs, or newlines. **wc** also counts characters and lines; all three counts are reported by default.

For example, to count the number of lines, words, and characters in the file *textfile*, enter:

```
wc textfile
```

Typical output describing lines, words and characters might be:

```
4432   18188    97808 textfile
```

To specify a count of characters, words, or lines only, you must use an appropriate mnemonic switch.

To illustrate, examine the following three commands and the output produced by each:

```
wc -c textfile
  97808 textfile

wc -w textfile
  18188 textfile

wc -l textfile
   4432 textfile
```

The first example prints out the number of characters in *textfile*, the second prints out the number of words, and the third prints out the number of lines.

4.9.6 Delaying a Process

The **at** program allows you to set up commands to be executed at a specified time. It is useful if you want to execute a command when you are not planning to be at your terminal, or even logged in.

The **at** command accepts standard input and has the form:

```
cat file | at time day
```

file is the name of the file that contains the command or commands to be executed. *time* is the time of day, in digits, followed by "am" or "pm." One- and two-digit numbers are interpreted as hours, three- and four-digit numbers as hours and minutes. More than four digits is not permitted.

day is optional. It is either a month name followed by a day number, or a day of the week. If no *day* is specified, the command will be executed today.

For example, if you want to find out what processes are running at 10 pm on Tuesday, place the following line in a file named *use:*

```
ps -a > /usr/myname/use
```

(See Chapter 2, "vi: A Text Editor," of the XENIX *User's Guide* for information on creating and inserting text into files.)

After you have written out the file and returned to command level, enter:

```
cat use | at 10pm tues
```

Press **RETURN**. The XENIX prompt reappears and you may continue working. At 10 pm on Tuesday, XENIX will execute **ps -a** and place the output in the file *use*. **at** is unaffected by logging out.

To check what files you have waiting to be processed, use the **at -l** command. **at -l** lists the files the user owns to be processed, along with the following information:

- The file's ID number
- The command invoking the file (**at** or **batch**).
- The date and time the file will be processed

To cancel an **at** command, first check the list of files you have to be processed and note the file ID number. Then use the **at -r** command to remove the file or files from the list.

The **at -r** command has the form:

```
at -r number
```

For example:

```
at -r 504510300.a
```

removes file number 504510300.a, canceling whatever commands were included in that file. A user can only remove his own files.

4.10 Controlling Processes

In XENIX, several processes can run at the same time. For example, you may run the **sort** program on a file in the "background", and edit another file in the "foreground" while the **sort** program is running. Things that you directly control at your keyboard are called foreground processes. Other processes, which you can initiate but that you otherwise have little control over, are called background processes. At any one time you can have only one foreground process executing, but multiple background processes may execute simultaneously. Controlling foreground and background processes is the subject of this section.

4.10.1 Placing a Process in the Background

Normally, commands sent from the keyboard are executed in strict sequence; one command must finish executing before the next can begin. Executing commands of this type are called foreground processes. A background process, in contrast, need not finish executing before you give your next command. Background commands are especially useful for commands that may take a long time to complete.

To place a process in the background, enter an ampersand (&) at the end of the command. For example, to count the number of words in several large files while simultaneously continuing with whatever else you have to do, enter:

```
wc file1 file2 file3 >count&
```

Output is collected in the file *count.* If output were not put in *count,* it would appear on the screen at unpredictable times as you continue with your work.

When processes are placed in the background, you lose control of them as they execute. For instance, entering **INTERRUPT** does *not* abort a background process. You must use the **kill** command, described in the following section, instead.

4.10.2 Killing a Process

To stop execution of a foreground process, press your terminal's **INTERRUPT** key. This kills whatever foreground command is currently running. To kill all your processes executing in the background, enter:

```
kill 0
```

To kill only a specified process executing in the background, first enter:

```
ps
```

ps displays the Process Identification Numbers (PIDs) of your existing processes, for example:

```
  PID TTY TIME CMD
3459 03  0:15 -sh
4831 03  1:52 cc program.s
5185 03  0:00 ps
```

In the above example, you might enter:

```
kill 4831
```

where 4831 is the PID of the process that you want killed.

Note

Killing a process associated with the **vi** editor may leave the terminal in a strange mode. Also, temporary files that are normally created when a command starts, and then deleted when the command finishes, may be left behind after a **kill** command. Temporary files are normally kept in the directory */tmp*. This directory should be checked periodically and old files deleted.

4.11 Getting Status Information

Because XENIX is a large, self-contained computing environment, there are many things that you may want to find out about the system itself, such as who is logged in, how much disk space there is, what processes are currently running. This section explains the types of information available from the system and how to get it.

4.11.1 Finding Out Who is on the System

The **who** command lists the names, terminal line numbers, and login times of all users currently logged on to the system. For example, enter:

```
who
```

This command produces something like the following output on your terminal screen:

```
arnold   tty02  Apr  7  10:02
daphne   tty21  Apr  7  07:47
elliot   tty23  Apr  7  14:21
ellen    tty25  Apr  7  08:36
gus      tty26  Apr  7  09:55
adrian   tty28  Apr  7  14:21
```

The **finger** command provides more detailed information, such as office numbers and phone extensions. For more information, about using **finger** see **finger**(C) in the XENIX *Reference Manual.*

4.11.2 Finding Out What Processes Are Running

Because commands can be placed in the background for processing, it is not always obvious which processes you are responsible for. The **ps** command stands for "process status" and displays information about currently running processes associated with your terminal. For instance, the output from a **ps** command might look like this:

```
  PID  TTY  TIME  CMD
10308  38   1:36  ed  chap02.man
   49  38   0:29  -sh
11267  38   0:00  ps
```

The PID column gives a unique process identification number that can be used to kill a particular process. The TTY column shows the terminal that the process is associated with. The TIME column shows the cumulative execution time for the process. Processes can be killed using the **kill** command. See section 4.10.2, "Killing a Process," for information on how to use the **kill** command.

To find out all the processes running on the system, use the **a** option:

```
ps -a
```

To find out about the processes running on a terminal other than the terminal you are using, use the **-t** option and specify the terminal number. For example, to find out what processes are associated with terminal 13, enter:

```
ps -t13
```

For more information about **ps** and its options, see **ps**(C) in the XENIX *Reference Manual.*

4.11.3 Finding Out Lineprinter Information

You can find out the status of files you requested printed with the **lpstat** command. **lpstat** displays information on an individual file or on all your files waiting to be printed.

To find out the status of one file, you need to know the "request ID." When you make print requests using the **lp** command, you find a request ID displayed on your terminal screen. The request ID has the form:

printer–idnumber

printer is the name of the printer your file will be printed on (check with your system manager for the names of printers available to you) and *idnumber* is a unique number identifying your file.

To find out the status of a particular file, enter:

lpstat *request ID*

lpstat responds by displaying the date and time you made your print request and the number of characters remaining to be printed.

To find out the status of all your files waiting to be printed on the lineprinters, enter:

lpstat

lpstat responds by displaying the request IDs and status information for all your files.

You can find out what files are waiting to be printed on a particular printer by using **lpstat** with the **-p** option. This command has the form:

lpstat -p *printer*

lpstat responds by printing the request IDs and status information for all the files waiting to be printed on the named printer.

For more information on **lpstat** and its options, see **lpstat**(C) in the XENIX *Reference Manual.*

4.12 Using the Lineprinter

The XENIX lineprinter commands are easy to use and give you great flexibility when you want to print a file. With a few simple commands, you can print multiple copies of a file, cancel a print request, or ask for a special option on a particular printer. Since the XENIX lineprinter system is designed to be easily adapted to many different environments, check with your system manager to find out what lineprinters and printer options are available to you.

4.12.1 Printing Files: lp

To print copies of your files, you can use either the **lp** command or **lpr**. These commands are equivalent. The examples in this section use **lp**.

For example, to print one copy of a file named *memo*, enter:

```
lp memo
```

You can request that several files be printed. For example, to print three files named *memo*, *report*, and *letter*, enter:

```
lp memo report letter
```

When you make print requests, **lp** responds by displaying your "request ID" on your terminal screen. Your request ID might look like this:

```
pr4-532
```

The first part (pr4) is the name of the printer your file will be printed on. The second part (532) identifies your file. Should you later wish to cancel your print request or check its status, you will find it useful to remember your request ID. For more information on these tasks, see sections 4.12.3, "Canceling a Print Request," and 4.12.4, "Finding Out the Status of A Print Request: lpstat."

One copy of each file you named will be printed on the default destination printer on your system.

You can use **lp** with pipes and other commands. The command to paginate a file is **pr**. To paginate and print a file named *textfile*, enter:

```
pr textfile | lp
```

To sort, paginate, and print a file named *datafile*, enter:

```
sort datafile | pr | lp
```

4.12.2 Using lp Options

The **lp** command has several options to help you control the output from your printer.

You can specify the number of copies you want printed by using the number option, **-n**. For example, to print two copies of a file named *report*, enter:

```
lp report -n2
```

Another option, **-d**, specifies your file's destination, that is, which printer your file will be printed on. Check with your system manager for the names of the printers available to you. To have two copies of a file named *report* printed on a printer named *quick*, enter:

```
lp report -n2 -dquick
```

Other useful options include:

- **-c** Makes a copy of the files you are printing. This prevents you from inadvertently removing or changing the file before it is printed.
- **-m** Sends you mail telling you your file has been printed.
- **-o** Specifies printer options. For example, you may be able to request that your document be printed using 12 pitch type. Check with your system manager to see what options are available for each printer or groups of printers on your system.
- **-r** Removes your files after printing.

For more information on options available for the **lp** command, see **lp**(C) in the XENIX *Reference Manual*.

4.12.3 Cancelling a Print Request: cancel

You can cancel a print request. For example, to stop printing a file with a request ID of *laser-245*, enter:

```
cancel laser-245
```

The **cancel** command immediately stops the file from being printed, even if the printer has already begun the print request.

You can also use the **cancel** command to stop whatever is currently printing on a particular printer. With **cancel**, you can easily free up a printer to print the next file, or stop it from printing strange output without contacting your system manager.

For example, to cancel whatever file is currently printing on a printer named *slow*, enter:

```
cancel slow
```

If the file did not belong to you, mail will automatically be sent to the file's owner reporting that the print request was canceled.

4.12.4 Finding Out the Status of A Print Request: lpstat

To find out the status of your files waiting to be printed, enter:

```
lpstat
```

lpstat gives output similar to:

```
prt1-121     chrisw     450    Dec 15 09:30
laser-450    chrisw    4968    Dec 15 09:46
```

The first column shows the request ID for each of your files being printed; the second column is your login name. In the third column, the number of characters to be printed is shown, and the fourth column lists the dates and times you made your print requests.

To learn the status of a particular file, use the **lpstat** command with your request ID. For example, to find out the status of a file with the request ID of *daisy-256*, enter:

```
lpstat daisy-256
```

lpstat displays the status of that file only.

You can also request the status of various printers on your system by using the **-p** option or by giving the name of the particular printer you are interested in.

To find out the status of all the printers on your system, enter:

```
lpstat -p
```

To find out the status of a printer named *quick*, enter:

```
lpstat -pquick
```

lpstat displays the request ID and status information for each file currently waiting to be printed on the printer named *quick*.

For more information on **lpstat** and its options, see **lpstat**(C) in the XENIX *Reference Manual*.

4.13 Communicating with Other Users

Because the XENIX system supports multiple users, it is very convenient to communicate with other users of the system. The various methods of communication are described below.

4.13.1 Sending Mail

mail is a system-wide facility that permits you and other system users to send and receive mail. To send mail to another user on the system, enter:

```
mail joe
```

where *joe* is the name of any user of the system. Following entry of the command, you enter the actual text of the message you want to send. Entry of text is terminated by pressing **Ctrl-d**.

A complete session at the terminal might look like this on your screen:

```
mail -s "Meeting today" joe
There will be a meeting at 2:00 today
to review recent problems with the
new system.
Ctrl-d
```

Note the use of the **-s** switch to specify the subject of the message.

For practice, send mail to yourself. (This is not as strange as it might sound – mail to yourself is a handy reminder mechanism.) You can also send a previously prepared letter, and you can send mail to a number of people all at once. For more details, see Chapter 3, "Mail", of the XENIX *User's Guide*, and **mail**(C) in the XENIX *Reference Manual*.

4.13.2 Receiving Mail

When you log in, you may sometimes get the message:

```
you have mail
```

To read your mail, enter:

```
mail
```

A heading for each message is then displayed on your terminal screen. When you press **RETURN**, the contents of the first message are displayed. Subsequent messages are displayed, one message at a time, most recent message first, each time you press **RETURN**.

After each message is displayed, **mail** waits for you to tell it what to do with the message. The two basic responses are **d**, which deletes the message, and **RETURN**, which does not delete the message (so it will still be there the next time you read your mailbox). To exit mail, enter: **q**, for "quit." Other responses are described in the XENIX *Reference Manual* under **mail**(C).

4.13.3 Writing to a Terminal

To write directly to another user's terminal, use the **write** command. For example, to write to joe's terminal, enter:

```
write joe
```

After you have executed the command by pressing **RETURN**, each subsequent line that you enter is displayed both on your own terminal screen and on joe's. To terminate the writing of text to joe, enter a **Ctrl-d** alone on a line.

The procedure for a two-way write is for each party to end each message with a distinctive signal, normally (o) for "over"; when a conversation is about to be terminated use the signal (oo) for "over and out."

4.14 Using the System Clock and Calendar

There are several XENIX commands that will tell you the date and time, or display a calendar for any month or year you choose. The following sections explain these commands.

4.14.1 Finding Out the Date and Time

The **date** command displays the time and date. Enter:

```
date
```

The date and time are displayed.

4.14.2 Displaying a Calendar

The **cal** command displays the calendar of any month or year you specify. This command has the form:

```
cal month year
```

For example, to display the calendar for March, 1952 enter:

```
cal 3 1952
```

The result is:

```
     March 1952

 S  M Tu  W Th  F  S
                    1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

The month must always be expressed as a digit. To display the calendar for an entire year, leave out the month. The year must always be expressed in full; the command "cal 85" displays the calendar for the year 85, not 1985.

4.15 Using the Automatic Reminder Service

An automatic reminder service is normally available for all XENIX users. Once each day, XENIX uses the **calendar** command to examine each user's home directory for a file named *calendar,* the contents of which might look something like this:

```
1/23 David's wedding
2/9  Mira's birthday
3/30 Paul's birthday
4/27 Meeting at 2:00
9/1  Karen's birthday
10/3 License renewal
```

calendar examines each line of the calendar file, extracting from the file those lines containing today's and tomorrow's dates. These lines are then mailed to you to remind you of the specified events.

4.16 Using Another User's Account

You can easily access another user's files, regardless of the permission settings, with the **su** command. The **su** procedure resembles logging in, and you must know the other user's password.

For example, to become user Joe, enter:

```
su joe
```

and press **RETURN**. When the password prompt appears, enter Joe's password. To cancel the effect of the **su** command and return to your own account, press **Ctrl-d**.

4.17 Calculating

The **bc** command invokes an interactive desk calculator that can be used as if it were a hand-held calculator. A typical session with **bc** is shown below. Comments explain what action is performed after each input line.

```
/* This is a comment */
123.456789 + 987.654321     /* Add and output        */
1111.111110
9.0000000 - 9.0000001 /* Subtract and output */
-.0000001
64/8       /* Divide and output      */
8
1.12345678934 * 2.3    /* Note precision   */
2.58395061548
19%4       /* Find remainder  */
3
3^4 /* Exponentiation   */
81
2/1*2      /* Note precedence        */
4
2/(1*2)    /* Note precedence again */
1
x = 46.5  /* Assign value to x      */
y = 52.5  /* Assign value to y      */
x + y + 1.0000 /* Add and output        */
100.0000
obase=16  /* Set hex output base */
15  /* Convert to hex        */
F
16  /* Convert to hex        */
10
64  /* Convert to hex        */
40
255 /* Convert to hex        */
FF
256 /* Convert to hex        */
100
512 /* Convert to hex        */
200
quit/* Must type whole word */
```

Also available are scaling, function definition, and programming statements much like those in the C programming language. Other features include assignment to named registers and subroutine calling. For more information, see Chapter 5, "bc: A Calculator", XENIX *User's Guide*.

Index

Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X